This, Getters y Setters en clases de Javascript

R

rafaella-ballerini

01/07/2022

Clases

La orientación a objetos en Javascript no es nada nuevo, pero mientras otros lenguajes usaban sintaxis de clase, en Javascript no se usaba. Esto siempre fué un detalle importante que incomodaba a algunos desarrolladores, ya que dificultaba un poco la migración entre tecnologías con facilidad.

A partir de ECMAScript 2015 (ES6), las clases comenzaron a ser parte del lenguaje. Esto no cambió ni agregó funcionalidad, pero trajo una mejor organización del código, además de entrar en el patrón de otras tecnologías.

Vea la diferencia:

```
//sin la sintaxis de clases
var persona = {
```

```
    nombre: 'Gabriela',
    apellido: 'Ribeiro',
}

//con la sintaxis de clases
class persona {
    constructor(nombre, apellido){
        this.nombre = nombre
        this.apellido = apellido
    }
}

let gabriela = new Persona ('Gabriela', 'Ribeiro')
```

Imagina que necesitas aislar o proteger de alguna manera los elementos de estas clases. ¿Qué usar para esto?

This

Imagina que declaras una clase `profesor`, de la siguiente manera:

```
class Profesor {
    constructor(nombre, apellido, curso){
        this.nombre = nombre,
        this.apellido = apellido,
        this.curso = curso
    }
}

let joao = new Profesor('João', 'Souza', 'Lógica de programación')
```

Y dentro de esa clase querías agregar un método -que no es más que una función específica para ella- llamado `saludos`, que imprimirá en la consola un "Buenos días" a este profesor.

Para ello, necesitaremos combinar las propiedades `nombre` y `apellido` al imprimir el saludo. Aquí es donde entra **this**. Esta palabra proviene del inglés "esto", por lo que es posible entender que su uso se hace para dirigir estas propiedades a la clase en cuestión. De la siguiente manera:

```
class Profesor {
    constructor(nombre, apellido, curso){
        this.nombre = nombre,
        this.apellido = apellido,
        this.curso = curso
    }
    saludos(){
        console.log('Buenos dias ' + this.nombre + ' ' + this.apellido)
    }
}

let joao = new Profesor('João', 'Souza', 'Lógica de programación')
joao.saludos() //Buenos dias João Souza
```

Pero entonces, ¿por qué no podemos usar el nombre de la clase para esto? Es simple, imagina que hay una variable global con el mismo nombre que `profesor`. Si usamos `profesor.nombre` dentro de la clase, puede haber un código ambiguo, lo que causará problemas.

Getter

El getter, con la sintaxis **get**, está asociado con una función que se llamará cuando se acceda a la propiedad en cuestión y se solicite dinámicamente. Puedes usarlo para retornar el estado de una variable interna, sin usar métodos explícitamente. De la siguiente manera:

```
class Curso {
    constructor(materia, profesor, duracion){
        this.materia = materia,
        this.professor = profesor,
        this.duracao = duracion
    }
    get prof() {
```

```
        return this.profesor
    }
}

let poo = new Curso('Programación orientada a objetos', 'Rafaella', '1 semestr
console.log(poo.prof) //Rafaella
```

En este ejemplo, solo usamos el getter para retornar un valor que ya se había declarado de forma fija. ¿Qué pasa si ahora queremos retornar un valor dinámico, como un promedio de otras propiedades? Podemos hacerlo de la siguiente manera:

```
class Boletim {
    constructor(participacion, prueba, trabajo){
        this.participacion = participacion,
        this.prueba = prueba,
        this.trabajo = trabajo
    }
    get media() {
        return parseInt((this.participacion + this.prueba + this.trabajo) / 3)
    }
}

let boletimSemestral = new Boletim(8, 6, 7.5)
console.log(boletimSemestral.media) //7
```

Algunos puntos importantes a destacar para el uso de getters son:

- **Puede tener un identificador numérico o de string.**
- **No debe tener ningún parámetro.**
- **No se puede utilizar más de un getter para la misma propiedad, ni puede haber una propiedad común con el mismo nombre que el getter.**

Setter

A menudo se usan junto con getters, **setters** se usan para establecer valores para una propiedad específica.

```
class Alumno {
    constructor(nombre, curso, semestre){
        this.nombre = nombre,
        this.curso = curso,
        this.semestre = semestre
    }
    set nombreAlumno(nombreAlumno) {
        this.nombre = nombreAlumno
    }
}

let lucas = new Alumno('', 'Ingenieria', 5)
lucas.nombreAlumno = 'Lucas'
console.log(lucas.nombre) //Lucas
```

Entonces, en este caso podemos llamar al setter pasando un parámetro para cambiar el valor de la propiedad del `nombre` del estudiante.

Algunos puntos importantes a destacar para el uso de setters son:

- **Puede tener un identificador numérico o de string.**
- **Debe tener exactamente un parámetro.**
- **No puede tener la misma nomenclatura de propiedad y función.**

Ahora podemos acceder a las propiedades de un objeto (con getters) o cambiar sus valores (con setters).

```
class Alumno {
    constructor(nombre, curso, semestre){
        this.nome = nombre,
        this.curso = curso,
        this.semestre = semestre
    }
    get nombreAlumno(){
        return this.nombre
```

```
    }
    set nombreAlumno(nombreAlumno) {
        this.nombre = nombreAlumno
    }
}

let lucas = new Alumno('', 'Ingenieria', 5)
lucas.nombreAlumno = 'Lucas'
console.log(lucas.nombre) //Lucas
```

¿Quieres aprender más sobre Javascript? Accesa:

Curso Online de JavaScript: primeros pasos con el lenguaje Curso Online de JavaScript: Programación orientada a objetos

Este articulo fue adecuado para Alura Latam por: Jose Charris

Cursos de Front End

ARTÍCULOS DE TECNOLOGÍA > FRONT END

un solo pago de US$49,90

- 218 cursos
- Videos y actividades 100% en Español
- Certificado de participación
- Estudia las 24 horas, los 7 días de la semana
- Foro y comunidad exclusiva para resolver tus dudas
- Acceso a todo el contenido de la plataforma por 6 meses

¡QUIERO EMPEZAR A ESTUDIAR!

Paga en moneda local en los siguientes países

ANUAL

US$79,90

un solo pago de US$79,90